2

Neural Network Systems Techniques in the Intelligent Control of Chemical Manufacturing Plants

Sung Hoon Jung
Hansung University

Tag Gon Kim
Korea Advanced Institute of Science and Technology

Kyu Ho Park
Korea Advanced Institute of Science and Technology

2.1 Introduction

Neural networks have been widely used in many control areas [1, 2, 3, 4, 5]. However, as controlled systems have been more and more complex, no one control paradigm is enough to control especially where the controlled systems are complex hybrid ones composed of discrete event systems and continuous systems. In order to control such hybrid systems, intelligent control methodologies must be embedded into an integrated intelligent control system [6, 7, 8, 9, 10, 11]. This integration provides an intelligent system with some capabilities such as self-learning, self-planning, and self-decision making [7, 9, 10]. The structure of the integrated system must be well defined and constructed for getting synergy effect from all modules. Thus, the basic framework for constructing an integrated system is very important.

Recently, Zeigler [12, 13] introduced an *event-based intelligent control* paradigm based on the simulation theory for discrete event systems [14, 15]. This control paradigm is devised using a simulation formalism called *DEVS (discrete event system specification)*. The DEVS provides mathematically-sound semantics to specify operations of discrete event systems in a hierarchical, modular manner [14, 15]. Therefore, this control paradigm can be a good framework especially where the controlled processes are highly complex, such as chemical plants. With this framework, high-level modules, such as planners and schedulers, can be easily constructed and other modules, such as neural networks and fuzzy logic control, can be easily incorporated with hierarchy and modularity [16].

VISIT…

In event-based control, a internal controller has an event-based model of a controlled plant that characterizes the operational properties of the plant [12, 13]. Using this model, the event-based controller can simulate the state of transitions in the plant and can diagnose the operations of the plant using time constraints. We used a neural network model as the event-based model. This neural network model has the dynamics of the controlled plant with discrete levels for simulating operations of the controlled plant, as well as for decision outputs to control. This event-based control system, with neural network models, can control any type of plant such as continuous plants, discrete event plants, and hybrid plants composed of continuous and discrete event systems.

The neural network model maps the dynamics of a controlled plant so that it can automatically generate dynamics for control and diagnosis. However, the mapped dynamics of a neural network model can not be exactly the same as those of the original plant, owing to incomplete learning. When a plant has a saturation property, this incomplete learning results in serious problems. This is because of the output of a nonlinear dynamic plant, with saturation property, is very sensitive to small changes of its input values.

In [17], the authors proposed a neural model predictive control strategy combining a neural network and a nonlinear programming algorithm. The control performance may be considerably degraded when the predicted value differs greatly from the actual value. This is because the control input depends greatly on the predictive value and there are no methods to compensate the predictive error. The application of the method to highly nonlinear plants may be very difficult or even impossible.

To cope with this problem, our method partitions a block, between a current state and a target one, into several intermediate blocks. Then control inputs are repeatedly applied to the plant until the target state is reached. This provides the event-based controller with a state feedback mechanism of the conventional control. The plant output may be slightly erroneous owing to incomplete learning. For this problem, a state window is employed that provides a state tolerance about a steady state error. These two windows, namely the time window and the state window, make a cross-check area to check the state and time constraints. This scheme may be viewed as a combination of a time-based diagnosis mechanism in an event-based control system [12] and a state-based control mechanism in a neural network control system [18].

We experimented with a continuously stirred tank reactor (CSTR) plant using our event-based control system with a neural network mapping model. The CSTR plant is a chemical process that has strong non-linearity and complicated dynamics. Experimental results show relatively good control performance in spite of the strong nonlinearity and complicated dynamics.

2.2 Neural Network Construction for Event-Based Intelligent Control

This section describes the neural network construction method of a continuous system for event-based intelligent control. First, we briefly review the event-based control paradigm and state the neural network construction method next.

Brief Review of Event-Based Intelligent Control Paradigm

In an event-based control, the event-based model of a plant is specified by an event-based control DEVS [19]. This event-based control DEVS is a modified version of the discrete event system specification (DEVS) formalism [14, 15, 13]. An event-based control DEVS is defined as a 7-tuples [19]:

$$M = \langle X, S, Y, \delta_{\text{int}} \delta_{\text{ext}}, \lambda, ta \rangle \tag{2.1}$$

where

- X is the external input events set; and
- $S = B \times X$, where B is the finite set of elements, each called a boundary; and

- Y is the finite output events set; and
- δ_{int} is the internal transition function; and
- δ_{ext} is the external transition function; and
- λ: $S \rightarrow Y$ is the output function; and
- *ta*: $S \rightarrow R^{+}_{0,\infty} \times R^{+}_{0,\infty}$, i. e., $ta(s) = [r, r']$, where $r, r' \in R^{+}_{0,\infty}$ and $r \leq r'$.

In an event-based control, an event-based model specified by the event-based-control DEVS is composed of an input set, a state set, and an output set. These three sets correspond to control inputs, control points, and threshold outputs of the plant in event-based control, respectively. The internal and external transition functions for the event-based model provide the behavioral characteristics of the plant. Each function in the tuple represents constraints on the system dynamics. The output function maps the discrete event states to threshold-like outputs. The *time advanced function* differs slightly from the original DEVS formalism [15] owing to its need for a time interval. The minimum and maximum times of this time interval, called a *time window*, provide time tolerance that acts as the conventional controller's counterpart to state tolerance. That is, the event-based controller regards the control operation as correct if the target state arrives within the time window.

After the event-based controller has output the control input to the real plant, the controller waits for a sensor signal from the threshold sensors. If the signal arrives prior to the minimum time of a time window, then the event-based controller issues a "*too-early*" error. If the signal arrives after the maximum time of the time window, the controller issues a "*too-late*" error. Even if the signal arrives within the time window, the controller issues an "*unexpected-state*" error if the signal is different from the target state. In summary, the event-based controller regards the operation of the plant as correct only if the signal arrives within the time window and is the same as the expected planned target state. The event-based controller sends diagnostic information—*too-early*, *too-late*, and *unexpected-state*—to diagnostic personnel to diagnose errors. This event-based paradigm has two advantages over the traditional ones; the error informations produced by the event-based controller could be used for diagnostic purpose and the use of a time window provides the controller with robustness against expected values from sensors.

Figure 2.1 shows the architecture of an event-based controller. An event-based controller is composed of a control part and a diagnosis part. The control part is composed of a *goal-driven-planner* (GDP) and a *control-output-generator* (COG). The diagnosis part is made up of a *simulator* and an *event-based model*. These two parts are managed by the central controller which provides a main control algorithm.

The logic of the control is as follows:

- The event-based controller receives an external input X_c from threshold sensors, and sends control signals Y_c to a plant if no error is detected.
- If errors are detected, the event-based controller sends diagnostic information Y_d to the diagnoser.
- The central controller internally sends a *current state* signal to the GDP to get the next target state, then sends the received target state with the current state to COG to get control output.
- The controller then sends the received control output to the plant. At the same time, it sends this output to the simulator to simulate the plant behavior.
- The simulator simulates the plant using an event-based model and generates two outputs, i.e., expected outputs Y_s of the plant and time windows *TW*.
- With the reference of the outputs and time windows, the controller diagnoses the state of the plant.
- If the expected state of a plant is sensed within the time window, then the event-based controller regards the current state of this plant as being correct.
- Otherwise, an error is assumed to have occurred, and the controller invokes diagnoser functions to find where it has occurred.
- The event-based controller repeatedly performs the described control logic in accordance with the sensor readings.

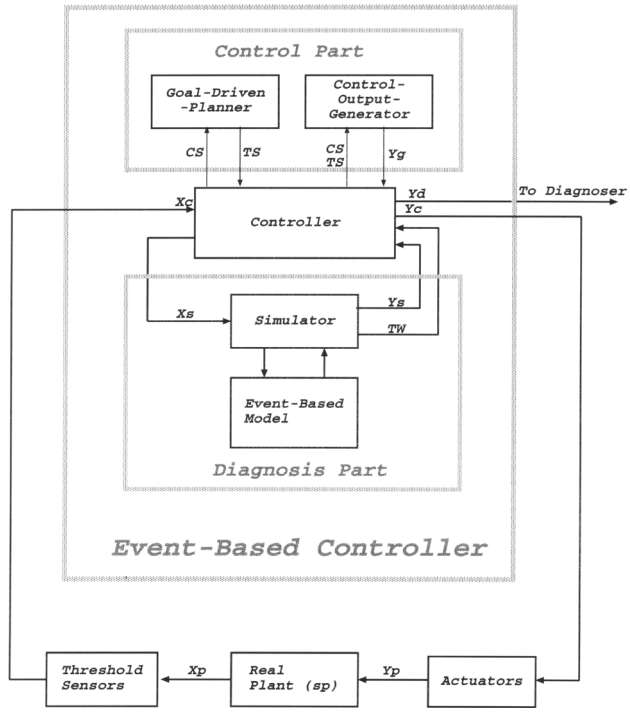


FIGURE 2.1 Event-Based Intelligent Control Environment.

Neural Network Construction Method

With mapping capability of neural networks, nonlinear plants could be easily modeled and these models have applied control in nonlinear plants in many different ways [1, 2, 4, 20]. Also, the on-line learning capability of neural networks makes it possible for control systems to have dynamic modelling capability. We use the mapping and on-line capability of neural networks.

In order to use a neural network model as an event-based model of an event-based intelligent controller, the controlled continuous system must be first abstracted into an event-based model and then the abstracted event-based model is mapped to a neural network. By doing this, the neural network model can be isomorphic to the continuous plant at the discrete input-output level [19].

Abstraction Process

Figure 2.2 shows the abstraction processes. The abstraction processes are as follows:

- When a continuous system is given, a system designer must first decide the operational objectives of the continuous system.
- The operational objectives can be derived from functional requirements and constraints of the system.
- To get the discrete inputs and outputs of the continuous system, the designer must do output partitioning first and input sampling second.
- The output partitioning is to divide the outputs of the continuous system into several mutually exclusive blocks to quantize the output levels considering operational objectives of the system.
- The input sampling is to select some inputs for state transitions specified by the operational objectives.

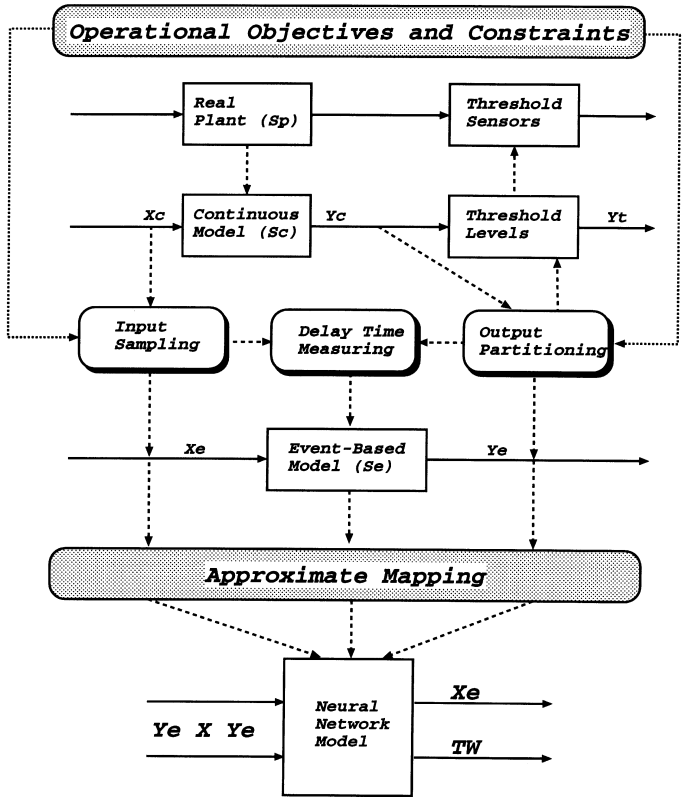


FIGURE 2.2 Abstraction Process and Neural Network Mapping.

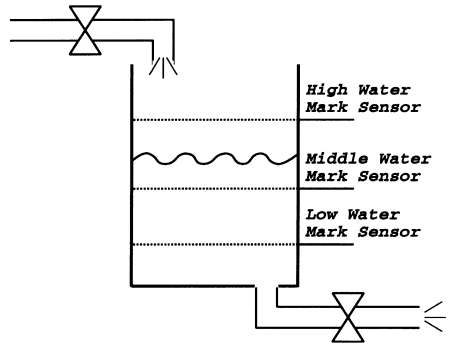


FIGURE 2.3 Water Tank Continuous System.

- Third delay times for the state transitions under given inputs must be measured using a real continuous system or a model such as a set of differential equations.
- With this information, we can construct an event-based model for the continuous system.

With a simple water-tank example, we illustrate the abstraction process. Figure 2.3 shows a water-tank continuous system. Let the constraints of the water tank system and operational objectives of this example

be given as follows:

- Constraints
  - The water tank must be able to supply the water to another device with two rates (not zeros).
  - The water can be supplied from another device with two rates (not zeros).
- Operational Objectives
  - The water level must be kept in the vicinity of the middle of the water-tank not to go over the top of the water-tank and not to go under the bottom of the water-tank.
  - Time to rise the water level from the bottom level to the middle level must not be over 2 minutes.
  - Time to fall the water level from the top level to the middle level must not be over 3 minutes.

From the first operational objective, we can do output partitioning with three levels: *High – Mark, Midd – Mark*, and *Low – Mark*. Of course, three threshold sensors for detecting the three levels must be equipped on the water-tank for real control. Using the second and third operational objectives and two constraints, we can next do input sampling. For simplicity, four values are selected for the input valve and output valve: *High – Input, Low – Input, High – Output*, and *Low – Output*. Of course, the symbolic elements for each set is assigned for real value: *High – Mark* = 3 m, *High – Input* = 10 lb/min and so on. Finally, delay times for state transitions, under given inputs, should be measured using a real continuous system or a differential equations model.

The water-tank operation is simply modeled using a first-order differential equation as follows.

$$C\frac{dh}{dt} = q_i - q_o \tag{2.2}$$

where C is the capacity of water-tank, h is the height of water-tank, q_i is the input flow rate of water, and q_o is the output flow rate of water. The height of water-tank is obtained by solving the equation:

$$h(t) = h_0 + \frac{q_i - q_o}{C}t \tag{2.3}$$

From this differential equation, the delay time can be calculated by the following equation

$$dt = \frac{h_c - h_0}{\frac{q_i - q_o}{C}} \tag{2.4}$$

where dt is delay time, h_c is a target state, and h_0 is a current state. Of course, the elements of h_c, h_0, q_i, q_o must be in the predefined set.

The dynamics of a real water tank will not be exactly the same as those of the differential equation model. This is caused not only by modeling errors but also by parameter changes of the real water tank and its environment. Let the minimum and maximum parameter variation be $P_{\min}$, $P_{\max}$ respectively, and the minimum and maximum times to reach a specific height h_c are as follows.

$$\begin{aligned} t_{\min} &= \frac{h_c - h_0}{\left(\frac{q_i - q_o}{C}\right)p_{\max}} \\ t_{\max} &= \frac{h_c - h_0}{\left(\frac{q_i - q_o}{C}\right)p_{\min}} \end{aligned} \tag{2.5}$$

TABLE 2.1 Real Values of States and Inputs of Water Tank

Water Mark			Two Input			
HIGH_TANK	MID_TANK	LOW_TANK	HIGH_IN	LOW_ IN	HIGH_ OUT	LOW_OUT
22.5	15.0	3.5	12.5	2.5	8.5	4.5

TABLE 2.2 Gathered Data from Differential Equation Model of Water Tank

Water Mark		Two Input		Time Window	
Current State	Target State	Input Rate	Output Rate	Minimum Time	Maximum Time
LOW_TANK	MID_TANK	HIGH_IN	HIGH_OUT	31.1	38.8
LOW_TANK	MID_TANK	HIGH_IN	LOW_OUT	15.5	19.4
LOW_TANK	MID_TANK	LOW_IN	HIGH_OUT	−20.7	−25.9
LOW_TANK	MID_TANK	LOW_IN	LOW_OUT	−62.1	−77.6
LOW_TANK	HIGH_TANK	HIGH_IN	HIGH_OUT	51.3	64.1
LOW_TANK	HIGH_TANK	HIGH_IN	LOW_OUT	25.7	32.1
LOW_TANK	HIGH_TANK	LOW_IN	HIGH_OUT	−34.2	−42.8
LOW_TANK	HIGH_TANK	LOW_IN	LOW_OUT	−102.6	−128.2
MID_TANK	LOW_TANK	HIGH_IN	HIGH_OUT	−31.1	−38.8
MID_TANK	LOW_TANK	HIGH_IN	LOW_OUT	−15.5	−19.4
MID_TANK	LOW_TANK	LOW_IN	HIGH_OUT	20.7	25.9
MID_TANK	LOW_TANK	LOW_IN	LOW_OUT	62.1	77.6
MID_TANK	HIGH_TANK	HIGH_IN	HIGH_OUT	20.2	25.3
MID_TANK	HIGH_TANK	HIGH_IN	LOW_OUT	10.1	12.7
MID_TANK	HIGH_TANK	LOW_IN	HIGH_OUT	−13.5	−16.9
MID_TANK	HIGH_TANK	LOW_IN	LOW_OUT	−40.5	−50.6
HIGH_TANK	LOW_TANK	HIGH_IN	HIGH_OUT	−51.3	−64.1
HIGH_TANK	LOW_TANK	HIGH_IN	LOW_OUT	−25.7	−32.1
HIGH_TANK	LOW_TANK	LOW_IN	HIGH_OUT	34.2	42.8
HIGH_TANK	LOW_TANK	LOW_IN	LOW_OUT	102.6	128.2
HIGH_TANK	MID_TANK	HIGH_IN	HIGH_OUT	−20.2	−25.3
HIGH_TANK	MID_TANK	HIGH_IN	LOW_OUT	−10.1	−12.7
HIGH_TANK	MID_TANK	LOW_IN	HIGH_OUT	13.5	16.9
HIGH_TANK	MID_TANK	LOW_IN	LOW_OUT	40.5	50.6

Consequently, we can get a time window by taking these minimum and maximum times:

$$t_{\min} \le t_{\text{win}} \le t_{\max} = \frac{C(h_c - h_0)}{(q_i - q_o)P_{\max}} \le t_{\text{win}} \le \frac{C(h_c - h_0)}{(q_i - q_o)P_{\min}}$$

Let the C, $P_{\min}$, and $P_{\max}$ be 12.2, 0.9037, and 1.12963 respectively, then the time window is given as:

$$\frac{10.8(h_c - h_0)}{(q_i - q_o)} \le t_{\text{win}} \le \frac{13.5(h_c - h_0)}{(q_i - q_o)} \tag{2.6}$$

These gathered data is mapped to the neural network with a back-propagation algorithm.

Let the output states (level of the water) and inputs (flow rates of input and output valves) be partitioned and sampled as shown in Table 2.1. To satisfy the first operational objective and two constraints, the high input rate of water to fill the water tank must be greater than the high output rate not to be emptied the water in the tank. Also, the low input rate of water to sink the water must be lower than the low output rate not to be overflowed the water. We can measure the minimum and maximum times for each state transition under given inputs. Table 2.2 shows the gathered data from Eq. 2.6. In this table, the minus time means that the delay time is infinity. That is, the target state will never be reached under given inputs.

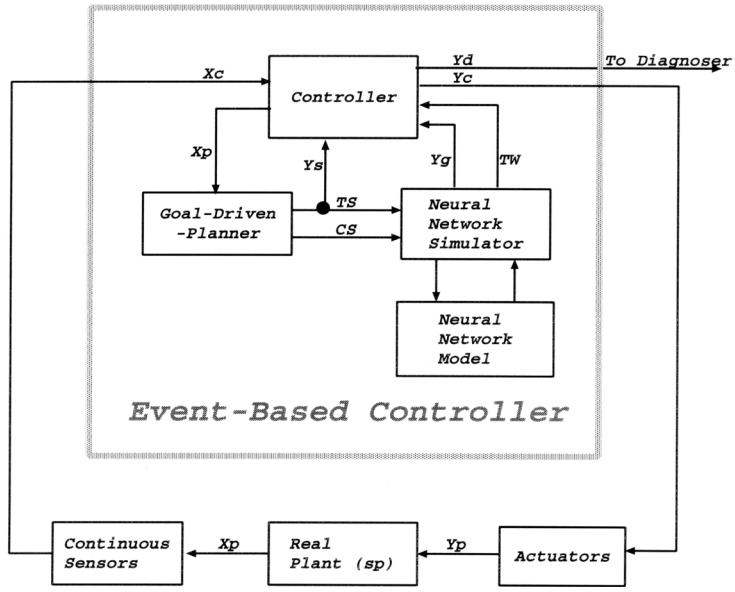


FIGURE 2.4 Event-Based Intelligent Control Environment with a neural network Model.

Neural Network Mapping

Figure 2.2 shows how the event-based plant model is mapped into a neural network. For mapping an event-based model to a neural network, the inputs and outputs of the neural network should be defined. The input and output events sets cannot be assigned for outputs of the neural network. This is because the neural network will generate outputs different from those of the training data owing to incomplete training. This makes it impossible to decide which event of the input and output events set are matched to the outputs of the neural networks. However, time windows of state transitions can be assigned for the output of the neural network because the time windows are able to have any real values in $R^{+}_{0,\infty}$ as shown in Eq. 2.1. Also control inputs can be assigned for the output of the neural network because the control inputs are not exactly the same as the sampled inputs. From this observation, we can decide the inputs and outputs of neural network. The inputs of neural network are composed of a current state value and a target state value. With these two inputs, the neural network generates the time window for the state transition and control inputs. Of course, all values of states must be in predefined state sets as elements. Note that the values of the elements for the event-based model are approximately mapped into a neural network model. Data not learned is generated by the neural network model in an approximate form. The structure of an event-based controller with an neural network model is shown in Fig. 2.4.

Model-Plant Mismatch Caused by Incomplete Learning

A back propagation algorithm, which is a typical supervised learning algorithm, is used for learning in our neural network model. The dynamics of an neural network model may not be exactly the same as those of the original continuous system owing to incomplete learning. This mismatch may cause the neural network model to generate inaccurate values. In the case of monotonically increasing or decreasing systems, the time window offers a tolerance against inaccurate time of state changes in the plant and uncertainty in the environmental changes. Initial state and plant parameter variations are examples of such inaccuracies and uncertainties [19].

However, this incomplete learning causes serious problems in the case of a saturated plant. This is because the output of a nonlinear saturated plant is very sensitive to changes of input values. Thus, the saturated value of the controlled plant may be greatly different from that of the target state. That is, the output of a saturated plant may not reach the target state under given inputs. This is because the

control inputs generated by the neural network model no longer make the output of the plant to be saturated with the exact target state. This problem can happen, not only because of the model-plant mismatch, but also because of some other factors (such as external disturbances and environment changes). We call these three elements *perturbation factors*. These error factors will not result in a fatal problem in monotonically increasing/decreasing systems because the output of the controlled plant will eventually reach the target state.

Recently, Song and Park [17, 21] proposed a predictive control scheme based on a neural network model for control of highly nonlinear chemical plants. The proposed scheme combines a neural network for plant identification with a nonlinear programming algorithm for solving nonlinear control problems. The method first generates a one-step-ahead predictive value of a controlled plant using the predictive neural model and then calculates a control input using an optimization algorithm, SQP (*Successive Quadratic Programming*) module, with a predictive value and a desired output value. The method shows good performance when the predictive value is approximately equal to the actual value. However, the performance may be considerably degraded when the predicted value differs greatly from the actual value. This is because the control input depends greatly on the predictive value and there are no methods to compensate for error between the two. Finally, the plant may have a large steady state error due to the predictive error.

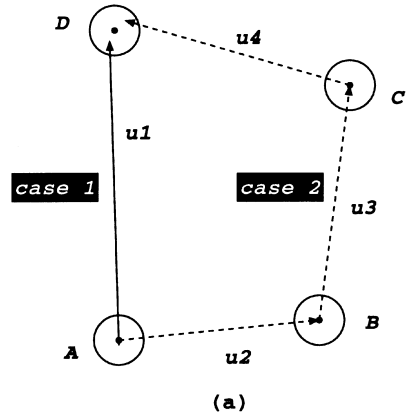
To cope with this problem, we partition the block between a current state and a target one into several intermediate blocks. Afterwards, we apply control inputs repeatedly to the plant until set points are reached. This provides the event-based controller with a state feedback mechanism used in conventional control. In fact, a control system that controls a nonlinear continuous plant does not work well without a state feedback mechanism. Thus, direct application of an event-based control system to control a nonlinear continuous plant may cause problems, such as those that occur in saturated plants. In this scheme, time windows are used to evaluate the saturated state of the plant, not to diagnose its state. That is, even though the state of the plant does not reach a target state within a time window, the controller does not generate the "*LATE*" error signal. Instead, it applies new inputs to the neural network model with a new current state.

This operation is applied to the plant repeatedly until the state of the plant reaches its target state. In spite of this operation, the plant output may be slightly erroneous, due to incomplete learning. That is, the neural network model may always generate the same control output Y_c for a specific current and target state.

This error can be seen as a steady state error which can be tapered by the continuous learning of the dynamics of a plant during control. To solve this problem, we adopt a state window—an interval of values of a state—which provides a state tolerance.

The width of the state tolerance accounts for the plant dynamics and the plant environment. These two windows—the time window and a state window—create a cross-check area that checks the state and time constraints with a tolerance. The event-based controller issues an error message only when the plant state is not within the cross-check area although n control operations are applied. The two dimensional error constraints of time and states are to provide the event-based controller with a 2-D maximum error tolerance in each direction. Consider the two situations shown in Fig. 2.5. In case 1, the output of a plant is saturated with a value within the error constraints although only one input u_1 is applied to the plant. This situation occurs when the current state and target state of the GDP are the same or slightly different from those of collected data. Even if the current state and target state are exactly the same as those of collected data, the output of a plant may not be exactly the same as the target state because the perturbation factors make the neural network model different from the collected data.

In case 2, the output of a plant is saturated with a value within the error constraints after receiving three inputs. This situation occurs when the planned current and target pair are not in the collected data, but in the transitive closure for the state transition such as $A \rightarrow B \rightarrow C \rightarrow D$ for $A \rightarrow D$ as shown in Fig. 2.5. This situation is also regarded as correct. Thus, the saturation of a plant output provides the event-based controller with the state feedback mechanism to avoid the saturation. In the two situations, the time constraints can be used to check the time requirements. However, if the target



CASE 1

CS	TS	Control Input
A	D	u1
.	.	.
.	.	.
.	.	.

(b)

CASE 2

CS	TS	Control Input
C	D	u4
A	B	u2
B	C	u3
.	.	.

(c)

FIGURE 2.5 Two cases of Control (a) state space (b) a data table (c) another data table.

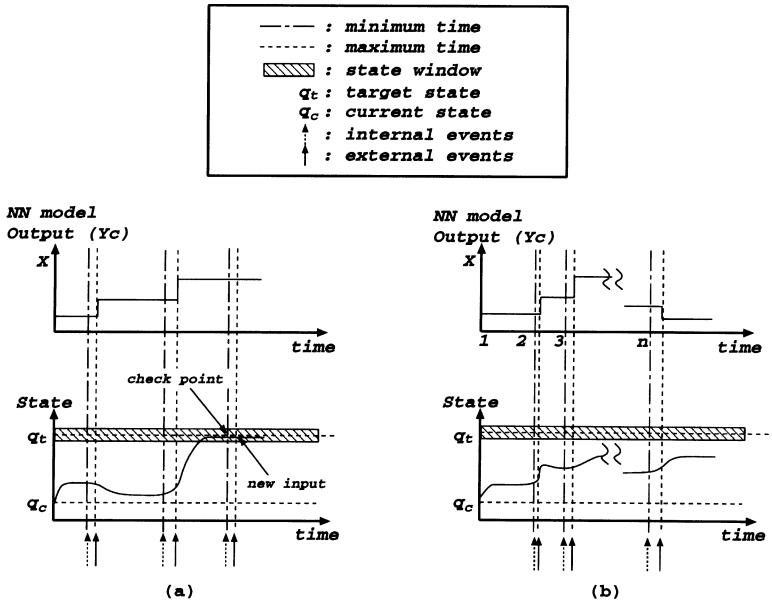


FIGURE 2.6 Two Control Situations of a Saturated Plant: (a) success case (b) error case.

state is not reached after n trials, then the event-based controller regards the state of plant itself or of environment as an error. The number n should be decided carefully in considering plant characteristics and operational constraints. This scheme may be viewed as a combined method of a time-based diagnosis mechanism of an event-based control system [12] and a state-based control mechanism of

a neural network control system [18]. Figure 2.6 shows the success case and the error case of the control operation.

Neural Network Model for Supervisory Control

Recently, supervisory control methods have been extensively researched [22, 23, 24, 25, 26]. A neural network model, represented by the above mentioned neural network modelling strategy, can also be used for supervisory control with small modification. In this application, the input/output flow rates of the water tank are assigned to inputs of a neural network model. This is because the supervisory control just allows some events in controllable events to occur for a state transition. The control scheme does not calculate the control inputs directly. Thus, the allowable control inputs (in other words, enabled controllable events) should be exactly defined and in a predefined set.

This is the reason why the control inputs of an event-based model cannot be assigned to outputs of a neural network model. With the simple water-tank example, we illustrate the application of a neural network model to the supervisory control. In supervisory control, the water tank model abstracted can be defined as a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, Q_m) \tag{2.7}$$

where

- Q is the state set, $Q = \{Empty, LOW\text{-}TANK, MID\text{-}TANK, HIGH\text{-}TANK, Over\ flow\}$
- Σ is the events set, $\Sigma = \{HIGH\text{-}IN, LOW\text{-}IN, HIGH\text{-}OUT, LOW\text{-}OUT, LOW\text{-}TANK\text{-}IND, MID\text{-}TANK\text{-}IND, HIGH\text{-}TANK\text{-}IND\}$
- δ: $Q \times \Sigma \rightarrow Q$ is the transition functions, for examples,
  - $\{LOW\text{-}TANK, (HIGH\text{-}IN, HIGH\text{-}OUT)\} \rightarrow MID\text{-}TANK$
  - $\{LOW\text{-}TANK, (LOW\text{-}IN, HIGH\text{-}OUT)\} \rightarrow Empty$
  - $\{HIGH\text{-}TANK, (LOW\text{-}IN, HIGH\text{-}OUT)\} \rightarrow MID\text{-}TANK$
  - $\{HIGH\text{-}TANK, (HIGH\text{-}IN, HIGH\text{-}OUT)\} \rightarrow Over\ flow$
  - $\{HIGH\text{-}TANK, (HIGH\text{-}IN, LOW\text{-}OUT)\} \rightarrow Over\ flow$
- $q_0 \in Q$ is the initial state, q_0: $= LOW\text{-}TANK$
- $Q_m \subset Q$ is the maker states, Q_M: $= MID\text{-}TANK$

The events, *LOW-TANK-IND*, *MID-TANK-IND*, and *HIGH-TANK-IND*, are indicative events of the three states, respectively. In this model, let the controllable events and uncontrollable events be given as:

$$\begin{aligned} \Sigma &= \Sigma_c \cup \Sigma_{uc} \\ &= \{HIGH\text{-}IN, LOW\text{-}IN, HIGH\text{-}OUT, LOW\text{-}OUT\} \cup \\ &\quad \{LOW\text{-}TANK\text{-}IND, MID\text{-}TANK\text{-}IND, HIGH\text{-}TANK\text{-}IND\} \end{aligned} \tag{1.8}$$

where Σ_c and Σ_{uc} are controllable and uncontrollable events, respectively. Then, a supervisor can drive from any states to the maker state because we select that any composition of flow rates of inputs and outputs can fill or sink the water.

In timed supervisory control, the abstracted neural network model can also be used for diagnosis of controlled systems. For example, when a supervisory controller enables two events, *LOW-IN*, *HIGH-OUT* and disables two events, *HIGH-OUT*, *LOW-OUT*, for state transition from *HIGH-TANK* to *MID-TANK*, the event *MID-TANK-IND* should occur within a time window generated from the neural network model. In all applications of abstracted neural network model, learning capability of neural network makes it possible for dynamic modelling of controlled system by on-line learning. This scheme is very

similar to human control strategy in that human can learn more and more information about the controlled system as the control actions proceed.

2.3 Simulation Environment

This section describes the controlled plant and the neural network learning strategy.

Continuously Stirred Tank Reactor (CSTR)

A CSTR is a chemical process that produces chemical products. The CSTR model is a part of a larger test system introduced by Williams and Otto [27] and McFarlane et al. [28]. The CSTR system, shown in Fig. 2.7, supports the following multiple reactions:

$$\begin{aligned} A + B &\xrightarrow{k1} C \\ C + B &\xrightarrow{k2} P + E \\ P + C &\xrightarrow{k3} G \end{aligned} \tag{2.9}$$

The desired product is P, while G, C, and E are byproducts subject to quality and environmental constraints. Reactants A and B enter as pure components in separate streams with flow rates F_{ai} and F_{bi}, respectively.

The flow rate F_{ai} and cooling water temperature T are variables in this chemical process. The input stream F_{bi} is considered to be a disturbance variable. The equations describing the kinetic behavior of the above reactions and the dynamic mass balance for the CSTR are a coupled set of nonlinear algebraic

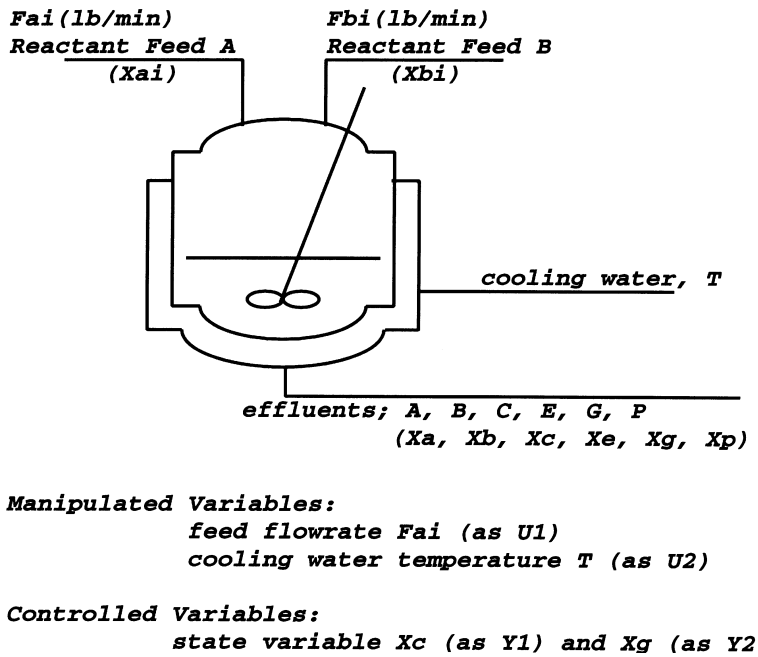


FIGURE 2.7 CSTR Plant.

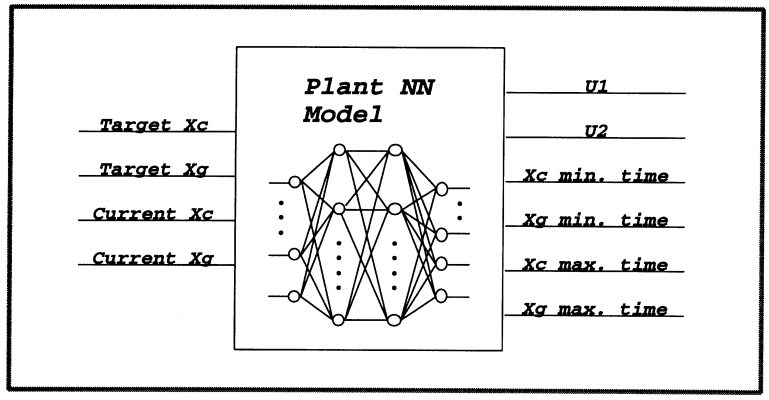


FIGURE 2.8 Neural Network Structure for CSTR.

and ordinary differential equations. A description of these equations has been provided by Williams and Otto [27]:

$$
\begin{aligned}
dXa/dt &= F_{ai}/Fr - rx1 - Xa \\
dXb/dt &= F_{bi}/Fr - rx1 - rx2 - Xb \\
dXc/dt &= 2rx1 - 2rx2 - rx3 - Xc \\
dXe/dt &= 2rx2 - Xe \\
dXg/dt &= 1.5rx3 - Xg \\
dXp/dt &= rx2 - 0.5rx3 - Xp \\
rx1 &= 5.9755E9 \exp(-12000/T)\, XaXb\rho V/(60Fr) \\
rx2 &= 2.5962E12 \exp(-15000/T) XbXc\rho V/(60Fr) \\
rx3 &= 9.6283E15 \exp(-20000/T) XcXp\rho V/(60Fr) \\
Fr &= F_{ai} + F_{bi} \\
\rho &= 50\,\text{lb/ft}^3,\ V = 60\,\text{ft}^3,\ 580^\circ R < T < 680^\circ R \\
X_i &= \textit{mass fraction}
\end{aligned}
\tag{2.10}
$$

The reaction constants, initial conditions, and constant parameters are as follows:

$$
\begin{aligned}
Xa &= 0.075 \quad Xe = 0.208 \quad F_{ai} = 170\ \text{lb/min} \\
Xb &= 0.57 \quad Xg = 0.0398 \quad F_{bi} = 679\ \text{lb/min} \\
Xc &= 0.015 \quad Xp = 0.019 \quad T = 645^\circ\text{R}
\end{aligned}
$$

The control objective of this system is to maximize the yield of the desired product P by regulating the two related state variables Xc and Xg. In this example, our neural network has 4-10-10-6 morphology as shown in Fig. 2.8.

Neural Network Learning Strategy

As shown in Eq. (2.1) of the event-based DEVS definition, the states of a continuous system are represented by the cross product of boundaries and inputs. Some initial control variable values can also be employed as initial states. If a system is represented by n boundaries, m inputs, and k initial states, then

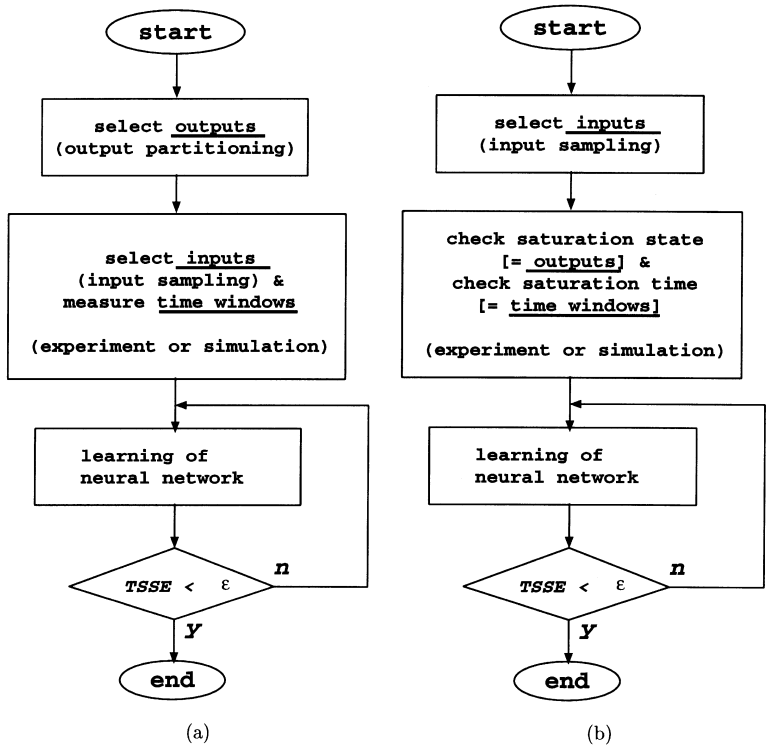


FIGURE 2.9 Flow Chart of State Space Mapping (a) original scheme (= in case of monotonic systems) (b) modified scheme (= in case of saturated systems).

$n \times m \times k$ number of state spaces are necessary to represent the system completely at discrete levels. For completeness, the states in the state space should be mapped to the neural network. For simplicity, we took only the input variables except for the initial states as independent variables.

Figure 2.9 shows the two flow charts of state space mapping algorithm. Figure 2.9(a) is the original state mapping scheme for monotonically increasing/decreasing systems. However, this scheme should be modified so as to fit with saturated plants as shown in Figure 2.9(b). First, in case of monotonic systems, control outputs should be determined using output partitioning. Then a sequence of control inputs should be selected such that the inputs move the current state of the plant to the target one. Finally, the control inputs are applied to the plant, and time windows measured. On the other hand, inputs within an operating range are first selected with random distribution for saturated plants. This is because it is difficult for the designer to find the inputs that make the output of the controlled plant saturated with a specific desired value. Thus initially, randomly selected inputs are applied to the plant and then the output of the plant is determined to be saturated or not.

The saturated state becomes an output state which can be used as training data. The saturation time is used to decide time windows. After collecting data, it is mapped to a neural network. To determine the mapping rate of the neural network model, a total sum square error (TSSE) is employed. The learning procedure is stopped only when the predetermined error bound, ϵ, is satisfied. As adequacy of the neural network model is dependent on ϵ, the value of ϵ should be carefully selected.

Algorithm 2.1 shows a detailed description of the collecting method of the mapping data.

Algorithm 2.1 Mapping-data-Collecting()
// N: the number of learning patterns //
1 **for** $i = 0$ to $N - 1$ // getting of N learning patterns //
2 select uniformly distributed random inputs within the operating range

```
3    record initial states
4    determine saturation time (Algorithm 2.2)
5    set collected data (Algorithm 2.3)
6  end for
```

The original mapping process from a plant to a neural network is shown in Fig. 2.2. However, this method cannot be applied to a plant that has a saturation property because the output partitioning does not affect the mapping process. For plants with a saturation property, inputs to the plant should be selected first; then the plant is determined to be saturated or not. If saturated, the saturation time is stored and used as a time window. That is, the target states are decided not from output partitioning in the original scheme but from the saturation property of the plants. Thus, the initial state and saturation state are taken to be the current state (*CS*) and target state (*TS*) respectively. In brief, the original method first determines the current state and target state with the output partitioning and then determines inputs with the input sampling. Afterwards, the length of time it takes for current state to reach it's target state is determined. Our method, on the other hand, first determines the inputs with the random sampling and then checks the saturated state with time. The saturated state and saturation time are regarded as a target state and a time window respectively.

The detailed description of the procedure for the determination of saturation time (time windows) is shown Algorithm 2.2.

Algorithm 2.2 Saturation-time-determination()

```
   // P: the number of plant output variables //
   // plant(): a simulation plant with discrete time-based simulation //
   // st(j): a saturation time of j'th output variable //
   // ct: a simulation time //
   // dt: a delta time //
1  do // determination of the saturation times //
2     for k = 0 to P − 1 // saving the previous plant outputs //
3        p_output_old(k) ← p_output(k)
4     for k = 0 to P − 1 // checking saturation //
5        plant (ct, initial(P), p_input(M), p_output(P) )
6        if p_output(k) = p_output_old(k) then
7           st(k) ← ct
8        end if
9     end for
10    ct ← ct + dt
11 until all p_output(P) = all p_output_old(P)
```

Although the neural network model represents the dynamics in a broader range, it cannot be exactly the same as the real plant itself. This is because the learning of the neural network is not complete. Thus, we add a Gaussian noise to the saturation time to provide a robustness against the external disturbances. Algorithm 2.3 shows details of the data setting procedure.

Algorithm 2.3 Data-setting()

```
   // N: the number of learning patterns //
   // M: the number of plant input variables /
   // P: the number of plant output variables //
1  for i = 0 to N − 1
2     for j = 0 to P − 1
3        nn_inp_pat(i, j) ← initial(j) // setting initial states //
4        nn_inp_pat(i, j + P) ← p_output(j) // setting target states //
5     end for
```

```
    for j = 0 to P − 1 // adding Gaussian noise //
        time1 ← st(j) ∗ gasdev()
        time2 ← st(j) ∗ gasdev()
        min_time(j) ← min(timel, time2) // determining time windows//
        max_time(j) ← max(timel, time2)
    end for
    for j = 0 to P − 1 // setting min. and max. times //
        nn_out_dat(i, j) ← min_time(j)
        nn_out_dat(i, j + P) ← max_time(j)
    end for
    for j = 0 to M − 1 // setting inputs //
        nn_out_dat(i, j + 2P) ← p_Input (j)
end for
```

The whole algorithm is outlined as follows. First, inputs within operating ranges are randomly selected with uniform distribution and are stored for later use. Second, current plant outputs are stored as initial states. Third, the selected inputs are applied to the plant, which is simulated with discrete time-based simulation. Finally, the time at which the output of the plant is saturated is measured and stored. Using this information, the state space of the plant is mapped into a neural network as an event-based model of the plant.

2.4 Simulation Results

Based on our event-based control scheme, we realized an event-based control system, HICON (High-level Intelligent CONtroller), with a neural network model of a plant on an expert system ART-IM/windows [29]. In this experiment, the CSTR plant was simulated using Eq. (2.10) with a discrete time based simulation method at 50 ms time intervals. The neural network plant model must complete learning the elements of the table model before the control operation begins.

Using the neural network model, a neural network manager generated the time windows and control outputs for the controller. New data obtained by the control operation can also be learned to dynamically adapt to the control environment. This dynamic learning provides flexible modelling capability to the event-based control system. Figure. 2.10 shows the overall controlled situation. The input/output states of the controlled plant are respectively depicted in the middle/bottom boxes of "CSTR Plant" window. In the output state, the desired set points are represented by bold lines and the controlled output by thin lines. As previously mentioned, the steady state error due to the incomplete learning is observed. This steady state error is generated only when the controlled plant has a saturation property. That is, when the plant output is saturated with respect to an input, model-plant mismatch causes the steady state error.

2.5 Conclusions

This chapter discussed a neural network application method for event-based intelligent control and supervisory control. We showed the usefulness of neural network modeling through an experiment of a chemical plant with a saturation property. This control method can be applied to any type of systems even hybrid systems composed of discrete event and continuous systems. In application to supervisory control, its diagnosis capability will enhance the performance of the supervisory control. The scheme may be viewed as a combined method of time-based diagnosis and state-based control. Experimental results showed that this scheme could be used to control a complex nonlinear plant and could be associated with the higher knowledge-based task management modules to construct more autonomous control system in further works [30, 31, 16].

Acknowledgments

The authors would like to thank Dr. J. J. Song for his helpful suggestions.

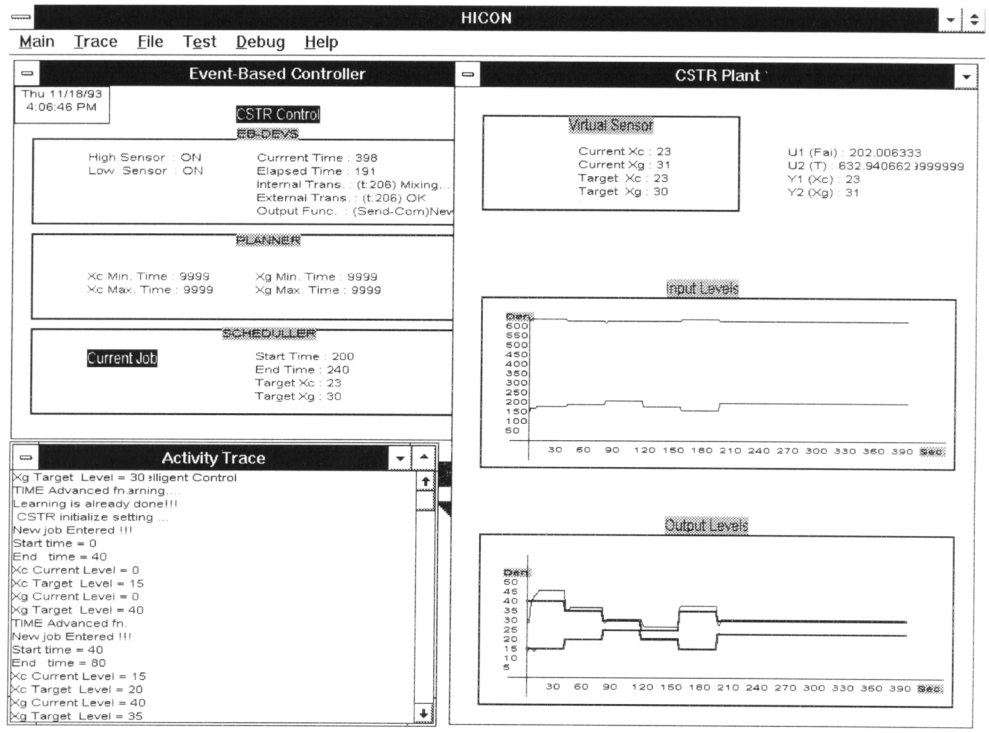


FIGURE 2.10 Result of CSTR Plant Control.

References

1. K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks,* vol. 1, pp. 4–27, Mar. 1990.
2. Y. Ichikawa and T. Sawa, "Neural Network Application for Direct Feedback Controllers," *IEEE Trans. on Neural Networks,* vol. 3, pp. 224–231, Mar. 1992.
3. C.-C. Lee, "Intelligent Control Based on Fuzzy Logic and Neural Net Theory," *Proceedings of the International Conference on Fuzzy Logic,* pp. 759–764, July 1990.
4. C.-T. Lin and C. G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE Trans. on Computers,* vol. 40, pp. 1320–1336, Dec. 1991.
5. S. ichi Horikawa, T. Furuhashi, and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-Propagation Algorithm," *IEEE Trans. on Neural Networks,* vol. 3, pp. 801–806, Sept. 1992.
6. F. Highland, "Embedded AI," *IEEE Expert*, pp. 18–20, June 1994.
7. P. Antsaklis, "Defining Intelligent Control," *IEEE Control Systems,* pp. 4–5,58–66, June 1994.
8. R. Shoureshi, "Intelligent Control Systems: Are They for Real ?," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, pp. 392–401, June 1993.
9. B. P. Zeigler, "High Autonomy Systems: Concepts and Models," *Proceedings of AI, Simulation, and Planning in High Autonomy Systems,* pp. 2–7, Mar. 1990.
10. S. Chi, *Modelling and Simulation for High Autonomy Systems*. PhD thesis, University of Arizona, 1991.
11. S. H. Jung, *Multilevel, Hybrid Intelligent Control System: Its Framework and Realization*. PhD thesis, KAIST, Feb. 1995.
12. B. P. Zeigler, "DEVS Representation of Dynamical Systems: Event-Based Intelligent Control," *Proceedings of the IEEE*, vol. 77, pp. 72–80, Jan. 1989.

13. B. P. Zeigler, *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*. Academic Press, 1990.
14. B. P. Zeigler, *Theory of Modelling and Simulation*. John Wiley & sons, 1976.
15. B. P. Zeigler, *Multi-Faceted Modelling and Discrete Event Simulation*. Academic Press, 1984.
16. S. H. Jung, T. G. Kim, and K. H. Park, "HICON: A Multi-Level, Hybrid Intelligent Control System," *IEEE Trans. on Systems, Man and Cybernetics*. submitted to IEEE Trans. on Systems, Man and Cybernetics.
17. J. J. Song and S. Park, "Neural Model Predictive Control For Nonlinear Chemical Processes," *Journal of Chemical Engineering of Japan*, vol. 26, no. 4, pp. 347–354, 1993.
18. A. Benveniste and P. L. Guernic, "Hybrid Dynamical Systems Theory and the SIGNAL Language," *IEEE Trans. on Automatic Control,* vol. 35, pp. 535–546, May 1990.
19. C.-J. Luh and B. P. Zeigler, "Abstracting Event-Based Control Models for High Autonomy Systems," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, pp. 42–54, JANUARY/FEBRUARY 1993.
20. C. L. Giles, G. M. Kuhn, and R. J. Williams, "Dynamic Recurrent Neural Networks: Theory and Applications," *IEEE Trans. on Neural Networks*, vol. 5, pp. 153–155, Mar. 1994.
21. J. J. Song, *Intelligent Control of Chemical Processes Using Neural Networks and Fuzzy Systems.* Ph.D. thesis, KAIST, 1993.
22. P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization,* vol. 25, pp. 206–23O, Jan. 1987.
23. P. J. G. Ramadge, "Some tractable supervisory control problems for discrete-event systems modeled by buchi automata," *IEEE Trans. on Automatic Control,* vol. 34, pp. 10–19, Jan. 1989.
24. R. Kumar, V. Garg, and S. I. Marcus, "Predicates and prediate transformers for supervisory control of discrete event dynamical systems," *IEEE Trans. on Automatic Control,* vol. 38, pp. 232–247, Feb. 1993.
25. T. Ushio, "A necessary and sufficient condition for the existence of finite state supervisors in discrete-event systems," *IEEE Trans. on Automatic Control*, vol. 38, pp. 135–138, Jan. 1993.
26. J. S. Ostroff and W. M. Wonham, "A framework for real-time discrete event control," *IEEE Trans. on Automatic Control*, vol. 35, pp. 386–397, Apr. 1990.
27. W. T. I. and R. Otto, "A Generalized Chemical Processing Model for the Investigation of Computer Control," *Trans. Am. Inst. Elect. Engr.,* vol. 79, pp. 458–465, 1960.
28. C. McFarlane and D. Bacon, "Adaptive Optimizing Control of Multivariable Constrained Chemical Processes. 1. Theoretical Development, 2. Application Studies," *Ind. Eng. Chem. Res.*, vol. 28, pp. 1828–1835, 1989.
29. "ART-IM/Windows Programming Language Reference." Inference Corporation, 1991.
30. T. G. Kim and B. P. Zeigler, "AIDECS: An AI-Based, Distributed Environmental Control System for Self-Sustaining Habits," *Artificial Intelligence in Engineering,* vol. 5, no. 1, pp. 33–42, 1990.
31. T. G. Kim, "Hierarchical Scheduling in an Intelligent Environmental Control System," *Journal of Intelligent and Robotic Systems,* vol. 3, pp. 183–193, 1990.